Angewandte Mathematik und Optimierung Schriftenreihe
Applied Mathematics and Optimization Series
AMOS # 33(2015)

Ingmar Vierhaus, Armin Fügenschuh, Robert Gottwald, and
Stefan Grösser

Modern Nonlinear Optimization Techniques for Optimal Control of System Dynamics Models

Herausgegeben von der
Professur für Angewandte Mathematik
Professor Dr. rer. nat. Armin Fügenschuh

Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
Fachbereich Maschinenbau
Holstenhofweg 85
D-22043 Hamburg

Telefon: +49 (0)40 6541 3540
Fax: +49 (0)40 6541 3672

e-mail: appliedmath@hsu-hh.de
URL: http://www.hsu-hh.de/am

Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Print 2199-1928
Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Internet 2199-1936

Modern Nonlinear Optimization Techniques for Optimal Control of System Dynamics Models

Ingmar Vierhaus[1], Armin Fügenschuh[2], Robert Gottwald[1], Stefan Grösser[3]

[1] Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany

[2] Helmut Schmidt University/University of the Federal Armed Forces Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany

[3] Bern University of Applied Science, Schwarztorstrasse 48, 3007 Bern, Switzerland

Abstract: Several existing software packages for system dynamics offer optimization and control capabilities. They allow to change the behavior of the system by modifying the value of policy variables to obtain an optimal outcome or to fit the model to historic data. The mathematical methods used are, for instance, Box' or Powell's method developed in the 1960s. They are able to solve models with just a few variables. These variables are independent of time. We summarize modern mathematical optimization methods and demonstrate how they can be adapted to solve system dynamics based optimization and control problems. As a technical obstacle, we have to transform table data, which is normally interpolated by piecewise-linear functions, to smooth spline functions instead. Such functions have first and second order derivates which are necessary for the new optimization method. By using this, modelers can solve larger optimization problems with many more control variables. In particular, the policy variables can now also be a time-varying function. As test cases, we consider two classical system dynamics models: Market Growth and World2. In both models, we identify parameters that can be used to optimize the models' dynamics and identify objective functions that we use to measure improvements over known base and optimal runs in these models. We introduce a cost of intervention into the models, effectively introducing a fixed budget for the possible interventions. We discuss the solutions and compare them and their objective function values to the base runs. In particular, we find a solution of the World2 model which significantly improves the published "towards a global equilibrium" run, while not incurring a higher cost of intervention.

Keywords: System Dynamics; Optimal Control; Nonlinear Optimization; Spline Interpolation.

Introduction

System dynamics models describe the behavior of dynamically complex social systems that consist of several interrelated stocks, flows, and feedback loops (Richardson, 2011). System dynamics models are mainly developed, first, to understand existing dynamics in systems, and second, to improve implemented policies, e.g., in project management (Ford and Sterman, 1998), to reduce the energy consumption in the residential built environment (Grösser, 2014), in city development (Forrester, 1969), in innovation management (Repenning et al., 2001), or in strategic management (Rahmandad and Repenning, 2015). System dynamics models consist of ordinary differential equations, nonlinear functional relations, and table data. Even if each of the elements in such a system is individually well understood, the interplay of several

of those elements may show a surprising, unexpected behavior over time (Forrester, 1971a). Given that one objective of system dynamics is to enhance current policies, we need to discuss how to compare two policies. When two simulations with different policies lead to different behavior of the system, one asks, which of the two simulations is more adequate or "better" for the given model purpose? To answer this question, one needs to define an objective function with respect to the purpose of the study. The objective function needs to be of such a kind that the higher the simulated value for the objective function, the better the policy (Dangerfield and Roberts, 1996). In this way, two different simulation runs and their underlying policies become comparable. Given that an adequate objective function has been defined, the next step is to identify the parameter configuration that results in the best possible value of the objective function, i.e., the best possible policy within the boundaries of the model. This is what is called "policy optimization". To identify an optimal policy, an optimization algorithm has to be deployed which can identify a local or even proven global optimal policy for a given situation.

In this article, we survey the approaches used in available system dynamics software packages. Two of the vendors use methods developed in the mid-1960s which are able to find local optimal solutions for optimization problems with few free variables. However, system dynamics models, especially models for real applications, often have several hundred or thousand variables and existing optimization algorithms are insufficient to handle these models efficiently. Recently, mathematical optimization algorithms such as interior point methods were developed that go beyond this limit. Such methods were already successfully applied to optimization problems having several thousand free variables and constraints (Wächter and Biegler, 2006). We demonstrate the applicability of these methods with two well-known system dynamics models as test cases. This requires the approximation of table data that are used to describe parts of the model by smooth curves that are twice continuously differentiable. To overcome this technical challenge, we use spline interpolation (De Boor, 1978). For both our test cases, numerical results are presented and discussed.

As mentioned, current system dynamics software packages perform optimizations with few free variables. Therefore, policy optimization is limited to a few constant (i.e., non-time varying) parameters. Our optimization approach has the potential to identify improved policies beyond what optimization methods currently in use in system dynamics can achieve. In particular, it becomes possible to provide decision makers with time-varying policy recommendations to address dynamic challenges. Pure parameter optimization (i.e., optimization of non-time varying parameters) is a special case of the optimization problems which we consider and can also be solved using the proposed method. This includes the calibration of a model to fit historic data.

Optimization Techniques for System Dynamics Models

We first review the best documented optimization techniques which are used by system dynamics software, which are Powell's method and Box' method. They are used in Ventana Vensim (Ventana Systems, Inc., 2015), Insight Maker (Fortmann-Roe, 2014a), and Goldsim (GoldSim Technology Group, 2015). There are other system dynamics software packages that also offer some optimization capabilities, for example, Powersim (Powersim Software AS, 2015), Anylogic (The AnyLogic Company, 2015), Dynaplan Smia (Dynaplan AS, 2015). However, these vendors do not reveal much details (if at all) about the method they use for optimization. Other software vendors do currently not include optimization, for example, isee systems' Stella (isee systems, inc., 2015) or Forio (Forio Corporation, 2015). In the Section "Interior Point Methods", we briefly describe interior point methods as an example for modern nonlinear optimization methods. Besides the mathematical founded optimization methods that we describe in the following, we remark that there are other non-mathematical heuristic improvement methods that were applied for an automated policy finding in system dynamics models, most prominently, genetic algorithms (Alborzi, 2008; Sholtes, 1994). For the calibration of models, Vensim also offers Monte-Carlo methods. In this paper, we will only consider local optimization techniques. Some remarks on the relationship to our work to global optimization can be found in the Section "Conclusions and Outlook".

Powell's Method

The method of Powell (Powell, 1964) finds a locally optimal solution (which is not guaranteed to be a global one), but for unconstrained optimization problems only, which are of the following form:

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_n) && \text{(1a)} \\ \text{s.t.} \quad & x = (x_1, \ldots, x_n) \in \mathbb{R}^n. && \text{(1b)} \end{aligned}$$

There are no further assumptions made on function f, in particular, it does not need to be a convex or a differentiable function. The method starts from an initial approximation of an optimal solution x^0. In each iteration, one tries to find a point that improves the objective function value (i.e., it has a higher objective function value) by performing a line search along n linearly independent search directions $\xi^1, \ldots, \xi^n$. Initially, these search directions are the coordinate vectors, and thus in the beginning one modifies one of the variables in $x^0 = (x_1^0, \ldots, x_n^0)$ at a time. In later steps of the method, the search directions are modified, such that the search directions are still linearly independent, but the search is not limited to coordinate vectors only.

Powell's method is usually applicable to small scale problems with up to 20-50 variables. To evaluate function f at a point x, the system dynamics model is simulated using the parameter set x. Powell's method is used as optimization method in the system dynamics software packages Ventana Vensim (Ventana Systems, Inc., 2015) and Insight Maker (Fortmann-Roe, 2014b). Its typical scope of application is parameter optimization and model calibration to historical data.

Box' Method

The method of Box (Box, 1965) deals with an optimization problem that can be stated in the following general abstract form:

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_n) && \text{(2a)} \\ \text{s.t.} \quad & g_k(x_1, \ldots, x_n) = 0, \quad \forall\, k = 1, \ldots, m, && \text{(2b)} \\ & x = (x_1, \ldots, x_n) \in \mathbb{R}^n. && \text{(2c)} \end{aligned}$$

In contrast to Powell's method, the method of Box is able to deal with constraints. Here $x_1, \ldots, x_n$ are free variables, f is the objective function, and g_k are the constraint functions. No assumptions on the smoothness of f and g are made, hence, they can be non-differentiable or non-continuous functions. It is only assumed that the feasible region $\{(x_1, \ldots, x_k) \in \mathbb{R}^n : g_k(x_1, \ldots, x_n) \leq 0\}$ is a convex set. In Box' method, an initial number of at least $n + 1$ points $x^1, \ldots, x^k \in \mathbb{R}^n, k \geq n + 1$ is sought that all fulfill the constraints $g_k(x^i) \leq 0$ for all $i = 1, \ldots, k$. Initially, these points are found using a pseudo-random generator that adds some noise to one initial solution x^0. The objective function f is evaluated for each point $x^1, \ldots, x^k$. The object $(x^1, \ldots, x^k)$ is called a complex. The point with the worst (smallest) objective function value is removed from the complex, and reflected at the centroid of the remaining points. The reflection is $\alpha \geq 1$ times as far from the centroid as the worst point, where α is a user-defined parameter. If the trial point is not feasible (i.e., it violates one of the $g_k \leq 0$ conditions), then some repair mechanisms take place. The procedure terminates, if the new points do not improve the objective function value over a certain number of rounds. Note that the method does not require smoothness of the constraints and objective function, or derivatives. Hence, it can easily be applied to system dynamics models with non-smooth functions arising from table data. Further note that the optimal solution is not a global or local optimal, but a feasible solution only. There is no guarantee that this method fulfills optimality conditions, hence in a mathematical sense, it might not even by a locally optimal solution. The problem sizes that can be handled with this method are usually rather small, around 20-50 variables and about the same number of constraints. Box' method is used in the system dynamics software package Goldsim (GoldSim Technology Group, 2015).

Interior Point Methods

In our work, we use state-of-the-art interior-point line-search algorithms, that were described by Wächter and Biegler (Wächter and Biegler, 2006) or Drud (Drud, 1985, 1994). These methods also deal with

constrained optimization problems of form (2). The method is based on the Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951), a necessary condition for local optimality of constrained optimization problems. These conditions yield a nonlinear equality system, which is solved multiple times during an optimization run for a parameter that converges to zero. When being (approximately) zero, a local optimal solution is found. This method, however, makes use of first and second derivative information, hence it must be assumed that all functions involved are at least twice continuously differentiable. As a positive effect of this, it can be applied to large problem sizes. Wächter and Biegler implemented their method in the publicly available software code IPOPT, and applied it to a test set of problems, where the largest has about 250,000 variables and constraints. A similar implementation based on a generalized reduced-gradient algorithm is available from Drud through the software code CONOPT, which we primarily use for our optimization computations, because our experiments revealed that CONOPT is – compared to IPOPT – more suitable for optimization problems with an inherent dynamic structure.

Treatment of Table Data and Table Functions

In many classical system dynamics models, table data is used to model functions, for example saturation curves or logistic functions (Franco, 2007). It is common practice to define such a function in terms of a table of points $(x_i, y_i), i \in \{0, 1, \ldots, n_P - 1\}$. The function value of a table function $f(x)$ is given as follows:

$$f(x) = \begin{cases} y_0 & \text{for } x < x_0 \\ (x - x_i)\frac{y_{i+1} - y_i}{x_{i+1} - x_i} + y_i & \text{for } x_i \le x < x_{i+1} \\ y_{n_P} & \text{for } x_{n_P} \le x \end{cases} \tag{3}$$

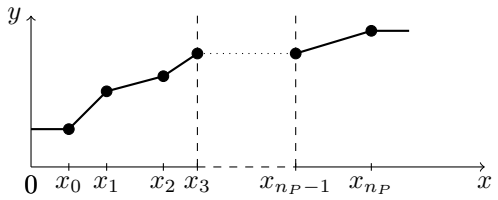


Figure 1: Example of a table function, given by the table data (points) $(x_0, y_0), (x_1, y_1), \ldots, (x_{n_P}, y_{n_p})$ and linear approximation in between.

For use in simulations, this linear approximation between discrete data points is an easy way to produce expected behavior. However, as described in Section "Optimization Techniques for System Dynamics Models", modern local optimization solvers rely heavily on evaluations of the first and second derivatives of a model's functions. The derivatives of a piecewise linear function are undefined at the points (x_i). Consequently, solvers like IPOPT or CONOPT cannot be used to solve system dynamics optimization problems, if a piecewise linear approximation of table functions is assumed. However, it is reasonable to assume that any set of discrete data points that is likely to appear in practice will stem from some continuous and, in most cases, smooth distribution of data. The non-smooth functions that are used to represent them, must then be considered an *artifact* of the representation, and not an inherent property. We claim that data that appears in system dynamics models can almost always be represented by a differentiable mathematical function without influencing the qualitative behavior of the model. This can be done by hand, but can also be successfully automated.

To assess the quality of an arbitrary smooth approximation, we need to first define a measure of the deviation between the original function $f(x)$ as defined in (3) and the approximation $\tilde{f}(x)$. The absolute

error err_{abs} and the relative error err_{rel} are given as follows:

$$\text{err}_{abs}(f, \tilde{f}) = \max_{x \in \mathbb{R}} \left| f(x) - \tilde{f}(x) \right| \tag{4}$$

$$\text{err}_{rel}(f, \tilde{f}) = \max_{x \in \mathbb{R}} \frac{\left| f(x) - \tilde{f}(x) \right|}{|f(x)|}. \tag{5}$$

The relative error gives a better measure of the potential change of the model behavior introduced by a approximation method, but it is undefined if the image of f includes the origin. We therefore introduce $\delta > 0$ and define a mixed error (Fügenschuh et al., 2015)

$$\text{err}_{mix}(f, \tilde{f}) = \frac{\left| f(x) - \tilde{f}(x) \right|}{f(x) + \delta}. \tag{6}$$

In the case $f(x) = 0$ it follows that $\text{err}_{mix} = \delta \cdot \text{err}_{abs}$, while in the limit $f(x) \to \infty$ we have $\text{err}_{mix} \equiv \text{err}_{rel}$. We will therefore adopt the mixed error as a reasonable compromise between absolute and relative error, and use it as the quality measurement for our approximation.

When it comes to the selection of our approximation method, smooth functions that preserve the shape of the table data are preferable, since artificial non-convex regions in the smooth approximations may have an adverse effect on the performance of the solvers. Local solvers might fail to move past these regions and hence converge to inferior solutions while global solvers might have to perform additional branching to prove global optimality (Vigerske, 2013).

The requirements for the functions calculated by the approximation algorithm are therefore as follows:

- $\tilde{f}(x), \tilde{f}'(x)$ are continuous
- The computational cost of evaluating $\tilde{f}(x), \tilde{f}'(x)$ and $\tilde{f}''(x)$ is similar to the cost of evaluating a piecewise linear function
- $\text{err}_{mix}(f, \tilde{f}) < \varepsilon$
- $\tilde{f}$ preserves the shape of f, i.e., $\tilde{f}$ is convex where f is.

To meet these requirements, we selected piecewise polynomial functions, also referred to as splines (De Boor, 1978). For these types of functions there are known methods for obtaining approximations that feature these properties.

The natural representation of a spline are the coefficients of its polynomial segments. However, for constructing a spline it is advantageous to represent it as a linear combination of basis functions. B-splines are such basis functions for the space of piecewise polynomial functions and have several properties that prove useful for the task at hand.

Evaluating a spline in either representation requires searching in a sequence of real numbers (to find the polynomial piece or non-zero basis functions). Once the spline interpolation has been determined, the computational effort of one evaluation is comparable to the evaluation of a piecewise linear function.

A B-spline basis is defined by a non-decreasing knot vector which controls the shape and continuity of the resulting curve. The continuity of the spline is decreased if the knot vector has non-unique components. At a knot position with k knots a spline of degree d is $d - k$ times continuously differentiable. We seek to obtain twice continuously differentiable functions, so we select cubic splines with a strictly increasing knot for our approximations.

Given the knots $t_1, \ldots, t_n$ and the function f, we use the so-called variation diminishing spline approximation to compute the coefficients $c_1, \ldots, c_k$ for the B-splines as follows:

$$c_1 = f(t_i^*) \quad \text{where } t_i^* = \left(\sum_{i=1}^{d} t_i \right) / d.$$

This method was chosen because it preserves the shape as well as the bounds of the function f, while maintaining sufficient approximation quality (De Boor, 1978).

With the above considerations, the spline approximation for a given function $f(x)$ and given knot positions $t_i \in \mathbb{R}, i \in \{0, 1, \ldots, n_{knots}\}$ in the B-spline space, is completely determined. For brevity, we denote the spline approximation of the source function $f(x)$ for given knot positions t_i as $\tilde{f}^{t_i}$.

Since $\mathrm{err}_{mix}(f, \tilde{f}^{t_i})$ depends nonlinearly on the knot positions and the knots are required to be strictly increasing, the problem of finding knot positions, such that the knots do not coalesce and $\mathrm{err}_{mix}(f, \tilde{f}^{t_i})$ is small, can be formulated as a constrained nonlinear optimization problem. In order to obtain an unconstrained problem, we use a parameter transformation of the knots (Jupp, 1978) which incorporates the monotonicity of the knot vector into the parameter space. We then employ nonlinear least squares methods (Zhou and Chen, 2010) to find a good placement of knots. Special care has to be taken in this step since almost equal knots can coalesce in finite precision which causes discontinuities in the spline. We address this issue in the line search of the nonlinear least squares method which does not accept steps where the knots are too close. The optimization of knot positions terminates if the change in $\mathrm{err}_{mix}(f, \tilde{f}^{t_i})$ was less than a fixed tolerance for a small amount of consecutive iterations or if the line search was not able to find an improving step anymore.

An outline of the approximation algorithm is given in Algorithm 1. The highest computational demand arises in line 6 of Algorithm 1, where the knot positions are determined. In the smooth approximations for the piecewise linear interpolations, the highest errors occur at the non-differentiable points. Thus, we decided to consider only the approximation errors at these points in the computation of the knot positions. This reduced the computational cost of this step substantially without negatively affecting the quality of the approximations.

Algorithm 1 Outline of approximation algorithm

Input: Table of points (x_i, y_i), defining $f(x)$ according to (3)
Input: Error tolerance ε
Output: Function $\tilde{f}(x)$ satisfying approximation requirements with tolerance ε

1: $\eta \leftarrow \infty$
2: $n_{knots} \leftarrow 3$
3: $\varepsilon \leftarrow$ error tolerance
4: **while** $\eta > \varepsilon$ **do**
5: $\quad n_{knots} = n_{knots} + 1$
6: $\quad \{t_i\} \leftarrow \arg\min_{\{t_i\}} \mathrm{err}_{mix}\left(f, \tilde{f}^{t_i}\right)$
7: $\quad \eta \leftarrow \mathrm{err}_{mix}(f, \tilde{f}^{t_i})$
8: **end while**
9: **return** $\tilde{f}(t_i)$

Automatic Generation of GAMS Models

The goal of applying the spline approximation technique, as described in the previous section, is to remove all non-smoothness from the considered system dynamics models. In the case of our two test models, this is successfull as all non-smooth functions can be approximated by smooth functions with very good accuracy (see Section "Application"). In our approach, the resulting smooth models are then extended to control problems and reformulated into nonlinear programs (NLPs). This reformulation allows us to solve the resulting optimal control problems with modern local NLP solvers, as described in Section "Optimization Techniques for System Dynamics Models". We used the GAMS (GAMS Development Corporation, 2013) modelling language to formulate the nonlinear problems.

This reformulation is for the most part straight forward. However, it is highly impractical to manually write a new GAMS model for every considered system dynamics problem. Writing each GAMS model from scratch is a tedious process, and requires manual changes to the GAMS model, whenever there is a change in the original system dynamics model. Also, there is no way to quickly change discretization schemes or parameters.

We recognized the need for a user friendly tool, which would allow to quickly convert control or parameter optimization problems based on system dynamics models available in the well-established Vensim format. As mentioned, Vensim has some optimization capabilities built in, and has already established a data format that allows to formulate parameter optimization problems based on system dynamics models. We extended this format to allow for the formulation of control problems. A control problem in this format is defined by four text files:

Model (`.mdl`) The model is supplied in the Vensim `.mdl` format and can be edited conveniently with Ventana's Vensim. Mathematically, the model contains a system of differential algebraic equations (DAE) for given parameters p and vectors x, y:

$$\dot{x}(t) = f(x(t), y(t), p) \tag{7a}$$
$$y(t) = g(x(t), y(t), p) \tag{7b}$$

Control (`.voc`) The control variables and ranges are defined in a `.voc` file, with a syntax that is compatible with Vensim parameter optimization problems. However, the format is extended to allow arbitrary and piecewise linear control variables.

Objective (`.vpd`) The objective is given in a `.vpd` file, again with Vensim compatible syntax, but with the additional possibility to easily define objective functions that integrate over the full time window for specified variables.

System Dynamics Optimization Problem (`.sdo`) The model, control definition and objective definition are linked via the `.sdo` file. This allows to use different objective functions or controls with the same model, by creating and linking a different `.vpd` or `.voc` file respectively.

The main input for our converter tool, which we call SDOCONV, is the `.sdo` file, which allows it to collect all information about the control problem from the respective files. Additional parameters like the discretization scheme can also be supplied to SDOCONV to change default behavior. The output is a smooth non-linear program, formatted as a GAMS model. The data flow is depicted in Figure 2. The basis of the reformulation is the explicit time discretization of the system consisting of (7a) and (7b). The variables x, y are no longer defined continuously on t, but only on certain equidistant time intervals given by the model parameters. We denote discrete times as bracketed superscript and rewrite the system (7a), (7b) as

$$\dot{x}^{(i)} = f\left(x^{(i)}, y^{(i)}, p\right), \quad \forall\, i \in \{0, 1, 2, \ldots, n\}, \tag{8a}$$
$$y^{(i)} = g\left(x^{(i)}, y^{(i)}, p\right), \quad \forall\, i \in \{0, 1, 2, \ldots, n\}, , \tag{8b}$$

where $n := \frac{T}{\Delta t}$ and $t^{(i)} := i\Delta t$ for all $i \in \{0, 1, 2, \ldots, n\}$. The resulting GAMS model then contains one variable for each state variable and algebraic variable at each time. Similarly, every algebraic or differential equation at each time is translated into one constraint in the GAMS model. SDOCONV also handles many of the built-in Vensim functions and reformulates them in an equivalent way that is valid in a nonlinear program. One exception are table functions, which are very closely approximated in the GAMS model.

SDOCONV is currently under development. A beta version has been published and can be found on github: `https://github.com/rgottwald/sdoconv`.

The GAMS modelling language supports extrinsic functions, that can be given through a callable library. We use this mechanism to supply the spline approximation to GAMS. To this end, a software package was developed. In order to run a model generated by SDOCONV that contains table functions, the lookup library needs to be available on the system. This library is also available as a beta version on github: `https://github.com/rgottwald/lookup`.

Application

We detailed our approach to the local optimization of system dynamics problems in the first part of this paper. We selected two models from literature to demonstrate the benefits of our novel optimization

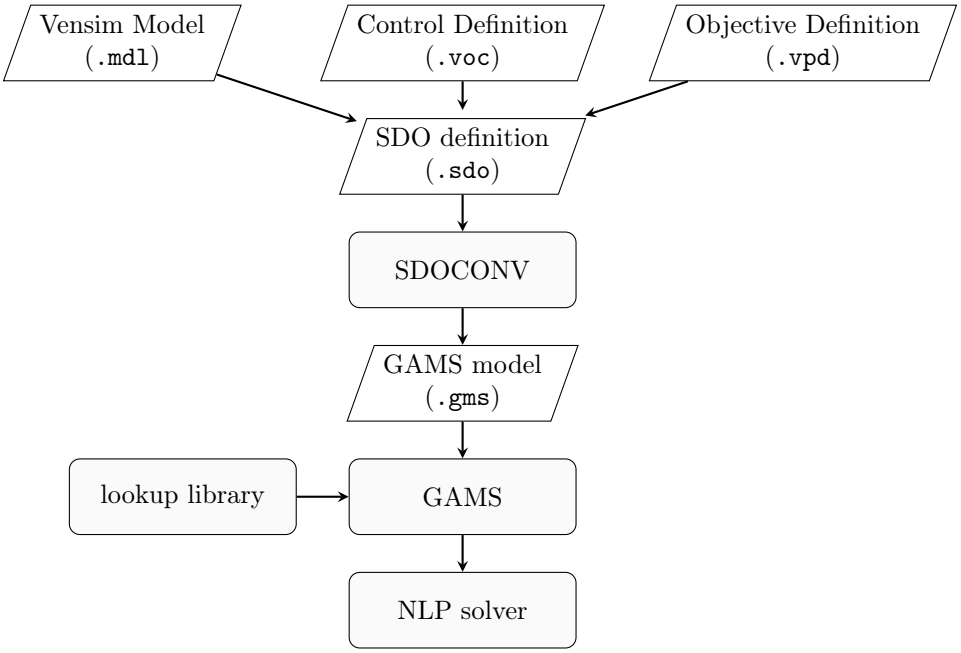


Figure 2: Data flow around SDOCONV

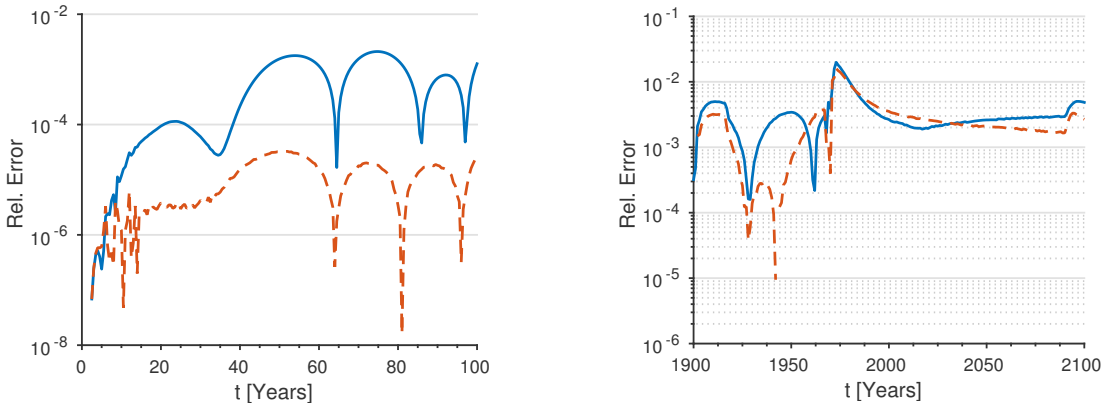
approach. The first is the Market Growth model from Forrester (1968); the second is the World2 model from Forrester (1971b).

When validating both models, one can reveal that both are highly robust against even extreme values. Both models have in common that today they have a status as generic or metaphorical models (Morecroft, 2012; Morecroft et al., 1995), i.e., they provide general insights for decision-making and do not rely on precise parameter values for the situation. This is why both models are also used for teaching at various academic institutions. The models are also similar with respect to their sizes, i.e., the number of stocks, variables, and resulting feedback loops. Each model consists of about five stocks and about 40 to 50 variables (incl. parameters). However, the resulting number of feedback loops for the stock variables differs significantly (about 12 for the market growth model and about 40 for the World2 model). Both models have the capability to assume very different behavior patterns and hence capture a high level of dynamic complexity (Grösser, 2012). Both models are professionally designed system dynamics models (Grösser and Tschupp, 2015; Rahmandad and Sterman, 2012) and both have been published by the originator of the field of system dynamics. The evidence rating (Homer, 2014) for both models we evaluate to be "A". This means that all non-obvious elements of structure and behavior are supported by multiple data sources and are critically examined. Moreover, both models are limited in their size (Ghaffarzadegan et al., 2010), and hence are ideal as "test models" to serve in our pilot study. In addition, the models have been extensively tested and manually optimized for the purpose of the original study. Hence, these models pose also a challenge for us to improve the best simulation runs with manually tuned parameters.

Spline Approximation Quality

We approximate all table functions in both models by splines, as described in Section "Treatment of Table Data and Table Functions". To verify that the spline interpolation does not lead to a change in the model behavior, we conducted three simulations of the base run of each model:

1. A simulation of the base run in Ventana Vensim, using the standard piecewise linear table function.
2. A simulation of the base run with a spline approximation, calculated with $\varepsilon = 10^{-2}$.
3. A simulation of the base run with a spline approximation, calculated with $\varepsilon = 10^{-3}$



(a) (b)

Figure 3: Comparison between base run simulations with piecewise linear functions and with spline interpolation. Plots show relative difference between simulation with piecewise linear function and an approximation with $\varepsilon = 0.01$ (solid lines) and an approximation with $\varepsilon = 0.001$ (broken lines). (a) The relative differences for the variable *Production capacity* in the Market growth model. (b) The relative difference for the variable *quality of life* for the World2 model.

We then proceeded to calculate the relative difference between simulations 1 and 2 and between 1 and 3 for a selected variable in both models at each point along the time axis. The results are shown in Figure 3. In Figure 3a, the result for the market growth model are shown. Even with the higher tolerance ε, the differences stay well below one percent. By choosing a higher tolerance, the accuracy can be improved further and then remains below 10^{-4} at all times. The World2 model contains a higher number of table functions, and the differences are slightly higher than in the Market growth model. However, the relative differences remain in the order of one percent or well below. We conclude from these experiments, that the computed spline approximations are a suitable smooth replacement of the table functions in both models. We will use the spline approximations computed with $\varepsilon = 10^{-3}$ for the computations in the following sections.

Controlling the Market Growth Model

First, we consider the Market Growth model as presented by Forrester (1968), a stock and flow diagram is shown in Figure 4. The model describes the policies governing the growth of sales and production capacity in a new product market. Forrester's original model resulted from a case study of an electronics manufacturer and represents the opinions of the company's senior management about the way that corporate growth was managed. We use the version of the Market Growth model as published by Richardson (2011).

In the well known base run of the Market Growth model, the evolution of the firm shows an unexpected stagnation. Over the first 40 months, sales rise, fueled by the increasing number of salesmen. Then, sales suddenly level off. This is because those sales are not being backed up with added production capacity; this produces delivery delays, starting around month 20, and begins to affect sales a few months after that. The development of the Production capacity and the Salesmen are shown in Figure 7.

In the original model, one of the key policies is modeled via the table function *CEF*, which defines the *Capacity expansion fraction* depending on the *Delivery delay condition*. Here, a production manager chooses how much the production capacity should be reduced or increased in the future. The argument of the function CEF, and therefore the basis for this decision, is the backlog of the company after some delay. The decision maker modeled here is subject to bounded rationality (Morecroft, 1985; Simon et al., 1984).

In our treatment of the model, we address the question whether it is possible to model a decision maker, who understands the dynamics of his company more comprehensively, instead of only locally. In other words, we question if a decision maker has the ability to account for more global information cues and overcome his or her locally bounded perspective and is, therefore, able to make more informed decisions.

And if the answer is yes, then we question, if there are alternative solutions to the optimization problem which allow the company to choose a policy which overcomes the decline of the production capacity in the considered time period.

The optimization problem resulting from these questions is detailed in the next section.

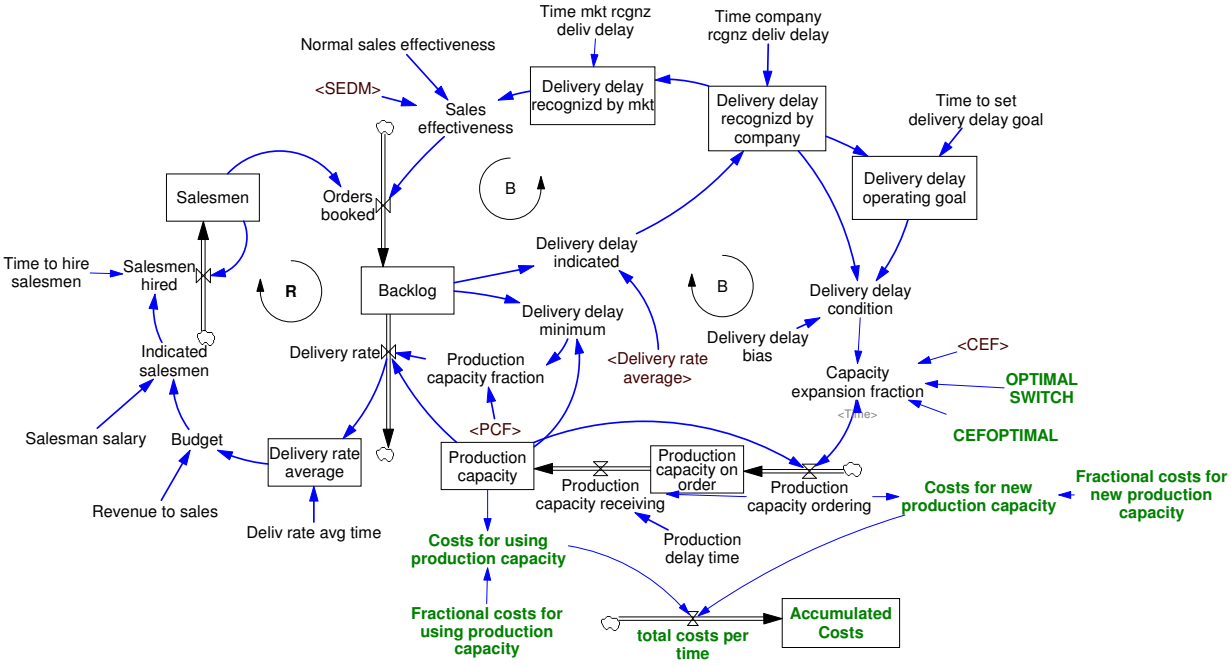


Figure 4: Structure of the adjusted Market Growth model (Forrester, 1968), source: (Richardson, 2011).

Problem Statement

As described in the previous section, the dynamics of the Market Growth model lead to a decline in the production capacity, as the company reacts to fluctuations in demand using the policies for capacity expension, hiring salesmen, and recognition of delivery delays. Our base model has the standard values for the policies as described by Forrester (1968). The key control the company has over the expansion or reduction of production capacity is in setting the variable *Capacity expansion fraction*. In the original model, this variable is defined in terms of a table function of the *Delivery delay condition*. From the original work of Forrester (1968), we know that a constant positive expansion of the production capacity will avoid the decline of production capacity and lead to a continuous increase. However, the original paper does not discuss, whether there is a cost associated with this expansion that might hinder a continuous constant expansion.

To be able to compare two simulations, we need to introduce a measure of the cost of an intervention. We therefore extend the original Market growth model by introducing two cost-variables $c_1(t), c_2(t)$, two weights w_1, w_2 and one new state variable integrating over the incurred cost.

$$c_1(t) = w_1 \cdot \text{Production capacity ordering}(t), \quad \forall\, t = 1, 2, \ldots, T, \tag{9a}$$

$$c_2(t) = w_2 \cdot \text{Production capacity}(t), \quad \forall\, t = 1, 2, \ldots, T, \tag{9b}$$

$$\text{Accumulated Costs} = \sum_{t=0}^{T} \Delta t \left(c_1(t) + c_2(t)\right). \tag{9c}$$

Note, that these additional variables are sinks. The dynamics of the other model variables are not changed by our modification. For the weights, we chose the values $w_1 = 8, w_2 = 0.5$. The first one, w_1, reflects the cost for creating new production capacity, w_2 reflects the cost for maintaining existing production capacity. We assume that one unit of new capacity is 16 times more expensive than mentaining one unit of capacity.

z_c	0.0001	0.005	0.001	0.002	0.05	0.01	0.1	0.2	0.4
Production Capacity(t) $\left[10^3\right]$	6.3	13.1	15.9	19.4	23.4	30.1	32.1	33.0	33.3

Table 1: Locally optimal objective values for the Market Growth model for different gradient limits z_c

In the base run, the accumulated cost incurred after 100 months amounts to a value of $3.47 \cdot 10^5$. In the following, we will consider two questions concerning the model:

1. Is there a capacity expansion policy with a similar cost of intervention that can keep the production capacity stable or even expand the production capacity within the considered time frame?

2. Can this capacity expansion policy be formulated as a function of the *Delivery delay condition*?

In the next section, we answer the first question by formulating and solving an optimization problem. By analyzing the solution of this problem, we will be able to answer the second question.

Formulation of the Market Growth Control Problem

At this point, the original model has been modified in two ways. The original non-smooth table functions have been replaced with highly accurate smooth interpolations, and we have added the cost accounting variables, as explained before, to asses a cost of intervention. For our optimization problem, we remove the functional dependency of the *Capacity expansion fraction* from the model and consider it a free variable. The table function CEF is thereby removed from the model. We refer to the equations of this new model with one free variable as "optimization model". We now formulate the following optimization problem:

$$\begin{aligned}
\max \quad & \text{Production capacity}(T), && \text{(10a)}\\
\text{s.t.} \quad & z(t) = \text{Capacity expansion fraction}(t), \quad \forall\, t = 1, 2, \ldots, T, && \text{(10b)}\\
& \text{Accumulated Costs}(T) \leq 3.5 \cdot 10^5, && \text{(10c)}\\
& \text{[model dynamics]}, && \text{(10d)}\\
& -0.2 \leq z(t) \leq 0.2, \quad \forall\, t = 1, 2, \ldots, T. && \text{(10e)}
\end{aligned}$$

Locally optimal solutions of control problems such as (10) often show so called "bang-bang behavior" (Artstein, 1980; Sonneborn and Van Vleck, 1965), i.e., the control remains at its upper bound for some time, and then switches abruptly to its lower bound and vice versa. Even though such a control can theoretically be very efficient, it can be impossible to realize in practice because the control cannot be changed infinitely fast. To take into account a finite changing speed of the control, we introduce additional variables z_c limiting the amount of change in the control from one time step to the next. These variables have the unit $\frac{\text{Unit of the control}}{time}$ and are incorporated into the model via constraints of the form

$$-z_c \leq \left(\frac{z(t+1) - z(t)}{\Delta t} \right) \leq z_c. \tag{10f}$$

We made experiments for different values of z_c. For a constant control ($z_c = 0$), no solution is found that respects the cost constraint. Therefore, we varied z_c from a value of 0.0001 (which leads to a very slowly changing control) to $z_c = 0.4$ (which allows the control to jump from one bound to the other within one time step and is therefore equivalent to no gradient limit).

The results are summarized in Table 1 and Figure 5. For the description in the sequel, we exemplarily select the value of $z_c = 0.002$ which leads to sufficiently slowly changing solutions for the Market Growth model. This constraint on the speed of change results in a reduction of 42% in the value of the objective function. The adjusted model is available in the online appendix.

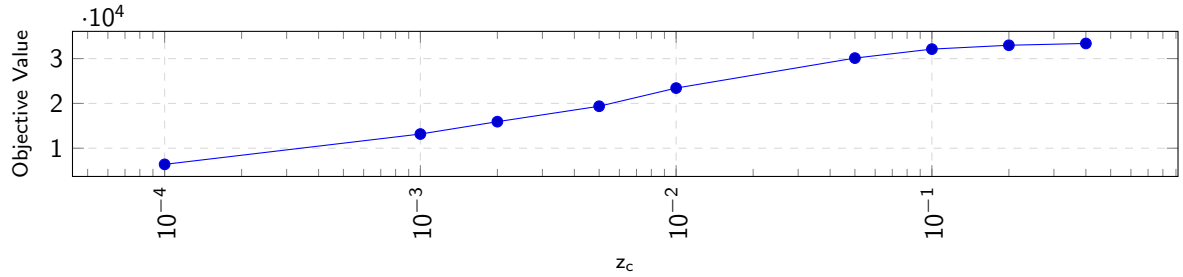


Figure 5: Plot of optimal objective values of the Market growth problem for different gradient limits z_c.

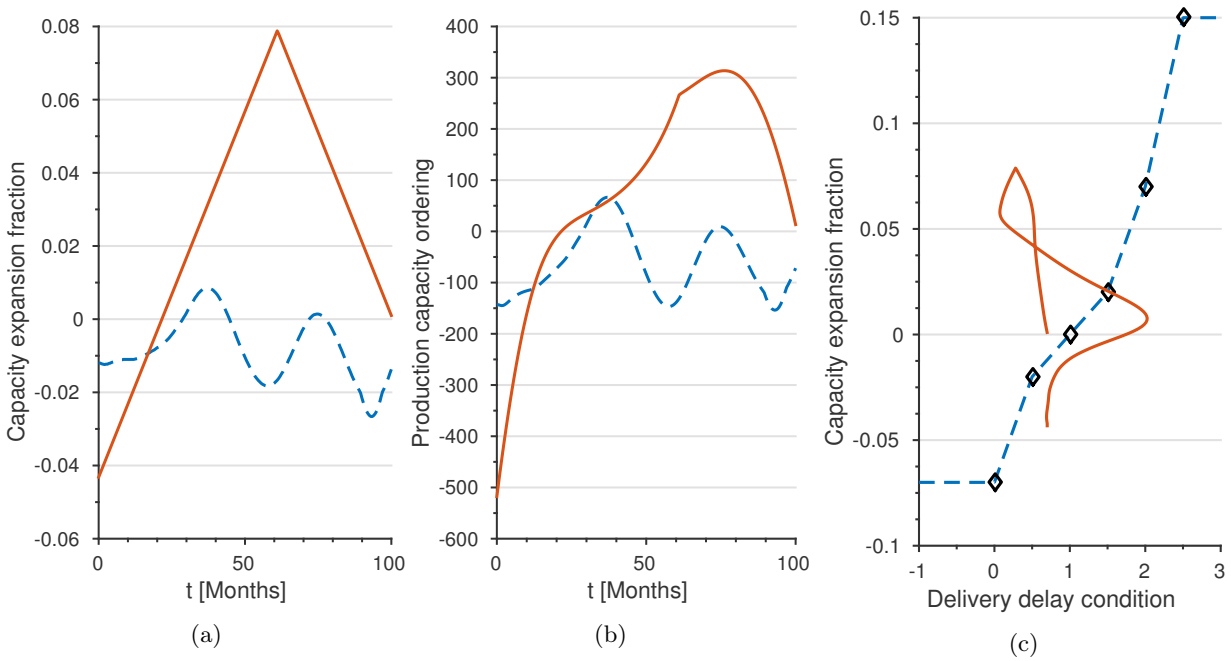


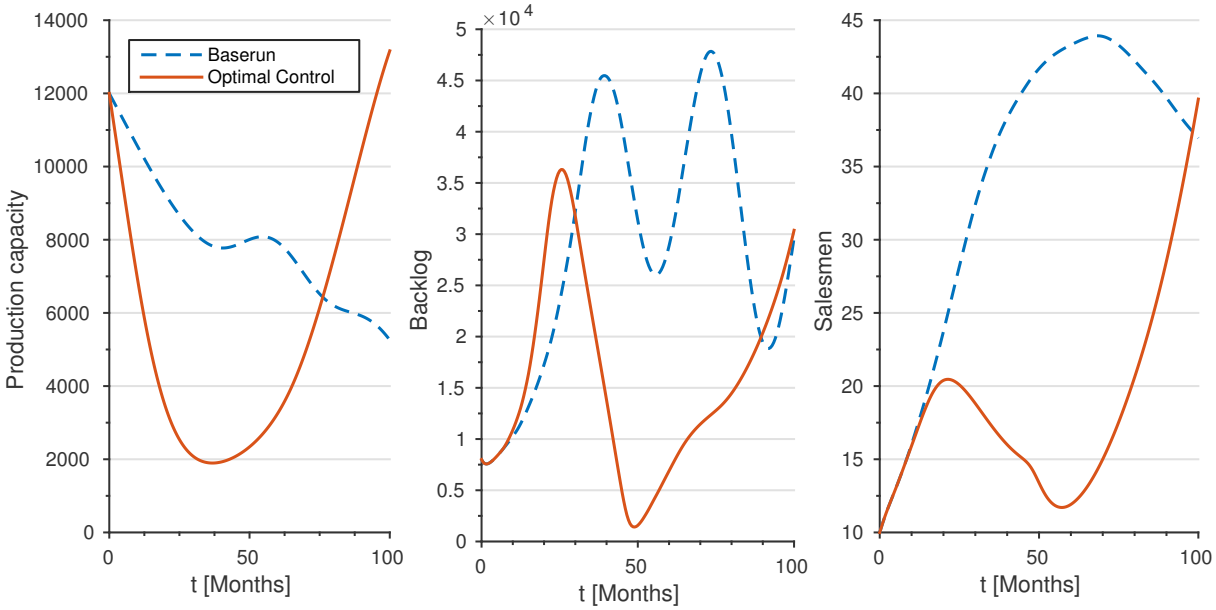
Figure 6: Broken lines show base run, solid lines show optimization solution. (a) Comparison of control variable in base run and optimal run. (b) Comparison of the ordered production capacity as a result of the chosen expansion fraction. (c) Plot of functional dependency between *Capacity expansion fraction* and *Delivery delay condition*.

Optimization results

We solve the problem using the NLP solver CONOPT (Drud, 1994). The single-core solving time on a workstation equipped with an Intel Xeon E3-1290 V2 was 4.3 seconds.

Figures 6 and 7 compare the solution from the optimization run with the base run. Figure 6a shows the control variable. In contrast to the base run, the optimal control starts with a strong reduction of production capacity, which is then gradually increased. In both runs, the first zero point is reached at almost the same time. However, while in the base run the first local maximum is reached shortly thereafter, in the optimized solution the variable continues to grow and reaches its maximum well in the second half of the considered time frame. The next figure shows the results of the control in terms of adding or reducing production capacity, and Figure 7a shows the actual *production capacities* over time. They show a regular u-shape ending with a higher production capacity at the end of the time frame than at $t = 0$.

With the above considerations, we can answer question 1 with "yes". We have found a satisfying solution, spending a similar cost of intervention, but avoiding the continuous decline of the production capacity. Indeed, as we can see in Figures 7b and 7c, backlog and the number of salesmen have also grown



(a) a) (b) b) (c) c)

Figure 7: Comparison of state variable values of the Market Growth model in base run and optimized run. Broken lines show the base run values.

within the time frame; hence, we do not have to expect a new strong oscillation after the considered time frame.

To answer question 2, we refer to Figure 6c. For the base run, we have plotted the table function CEF, i.e., the function has exactly one value on the second axis for each value on the first axis. In the optimal control case there is clearly no function of the *Delivery delay condition* that would produce this plot. The answer to question 2 is therefore "no".

Since in the optimization procedure, for each solution of the problem all variable values at all times are known, the formulation and solution of the optimization problem (10) can be interpreted as modelling a decision maker who is aware of the full model and its development over time. We have shown that it is impossible to formulate a table function of one argument, that reproduces this behavior. Whether it would be possible to calculate a multivariate function in the variable values, that would lead to a similar solution is an interesting question, and remains as a topic for future research.

Controlling the World2 Model

The World2 model was introduced by Forrester (1971b). Figure 8 shows its stock and flow diagram. The model is Forrester's answer to the futility of addressing world challenges in a piecemeal fashion. Instead the problem should be addressed as a system of problems. The model consists of five interacting subsystems, each of which deals with a different system of the model. The main systems were the food system, dealing with agriculture and food production, the industrial system, the population system, the non-renewable resources system, and the pollution system. The model shows that the production of goods, especially food, on this world is limited by the available resources and constraints which will prevent population and production from growing infinitely.

Problem Statement

In (Forrester, 1971b), several scenarios are considered for the World2 model. In particular, in Chapter 6 "Towards a Global Equilibrium" parameter settings are presented, that lead to a sustainable state of the world system within a relatively short time. We use this scenario as our base run. Selected state variables

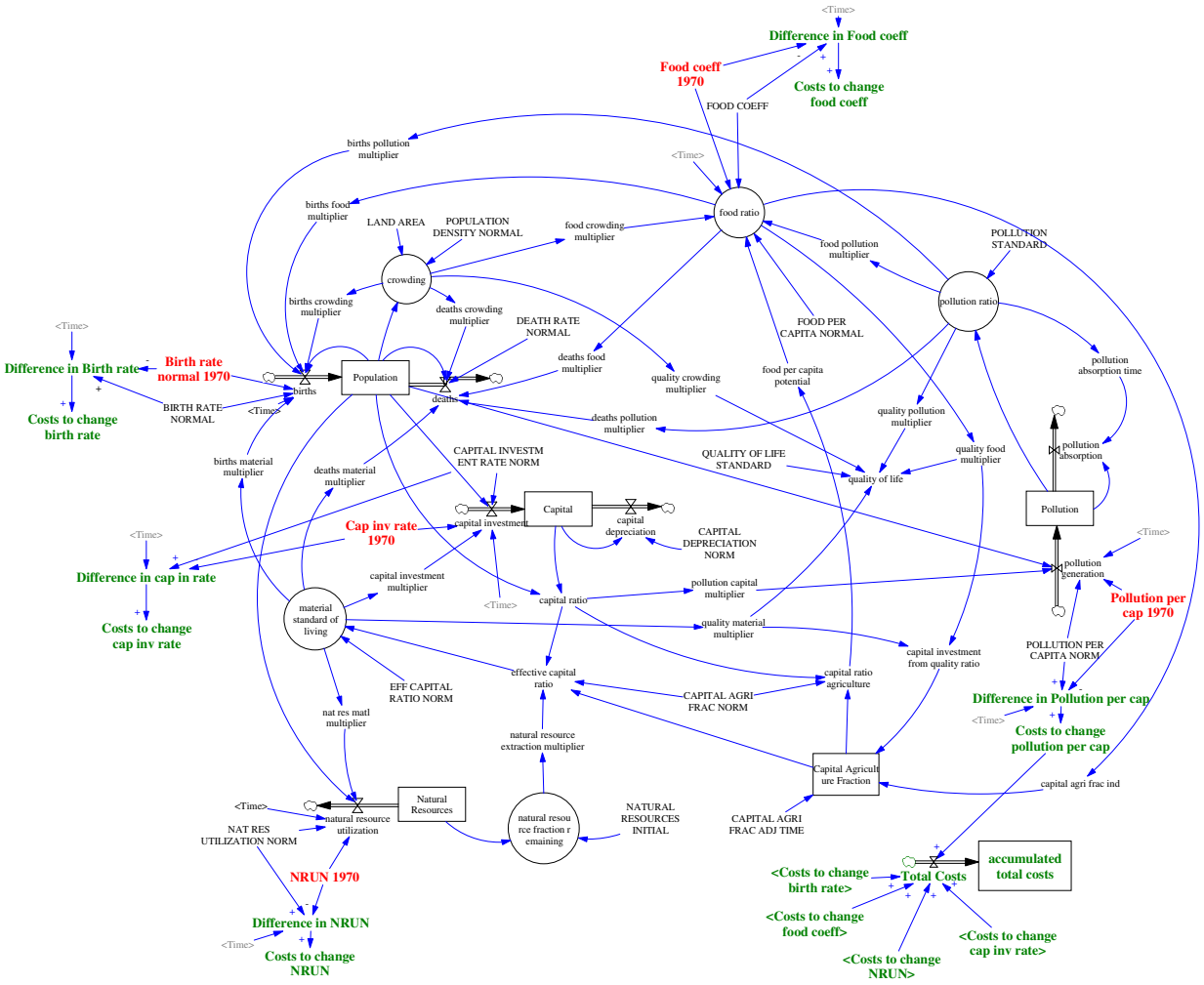


Figure 8: Model structure of the World2 model (Forrester, 1971b), source: teaching material of Georg Richardson, University of Albany, PAD 624).

of this run are shown in Figure 9 and the parameter changes suggested by Forrestser are listed in in Table 2 in the column "Base run". In the base run, we see a stabilized population level, a high level of quality of life and low level of pollution. For our optimization, we intend to improve the system behavior of the World2 compared to the best Forrester policy. We want to implement policies to achieve a more sustainable world measured in *quality of life* with the smallest costs necessary to achieve this. As in the previous model, we introduce costs for changing the policy variables (see the controls in Table 2 for the names of the variables and the selected values of the cost coefficients). The controls are: *NRUN* which accounts for the *natural resource utilization rate*, *pollution per capital*, *capital investment rate*, *food coefficient*, and *normal birth rate*.

As in the base run, all variables remain at their initial value z_{init} until the year 1970 and can only be changed afterwards. We selected the following symmetric exponential function, to model a cost that increases exponentially with increasing magnitude of the intervention. For each of the five controls i as given in Table 2, the cost incurred by a change from the initial value $z_{init,i}$ to a value z_i is then given by

$$f_{c,i}(z_i, z_{init,i}) = \alpha \left(\exp\left(\beta(z_i - z_{init,i})/w_i\right) - 1 + \exp\left(-\beta(z_i - z_{init,i})/w_i\right) - 1\right) \tag{11}$$

where we choose the parameter values $\alpha = 2.9 \cdot 10^4$ and $\beta = 3.6$. Each policy variable z_i must remain within a given interval (see Table 2. The weights w_i are normalizations for different interval width and ensure an adequate weighting of the individual cost components. The values for the weights are given in

Control z_i	i	Value $t < 1970$	Base run	Optimization Range	Weight w_i	$z_{c,i}$
NRUN1970	1	1	0.25	$[0.1, 1.0]$	0.9	0.045
Pollutionpercap1970	2	1	0.5	$[0.1, 1.0]$	0.9	0.045
Capinvrate1970	3	0.05	0.03	$[0.01, 0.05]$	0.04	0.002
Foodcoeff1970	4	1	0.8	$[0.6, 1.25]$	0.65	0.0325
Birthratenormal1970	5	0.04	0.028	$[0.02, 0.04]$	0.01	0.0005

Table 2: Controls for World2.

Table 2.

We sum the costs of changing the controls and accumulate them over time. The final cost of one run is then given as follows:

$$\text{Accumulated total costs(T)} = \sum_{t=0}^{T} \Delta t \left(\sum_{i=1}^{5} f_{c,i} \left(z_i(t), z_{init,i} \right) \right) \tag{12}$$

As in the market growth model, information flows only into our newly introduced bookkeeping variables. Therefore, the model dynamics are not changed by these modifications.

With the model adjusted as just described, and which is available in the online appendix, we re-simulated the model with Forrester's parameter choices and derived the costs incurred by his policy changes. This total accumulated cost component we call Forrester Budget, i.e., the costs necessary to implement Forrester's final policy for World2. The computed value is $3.8 \cdot 10^7$. We will use this Forrester Budget as a reference value for the optimizations in the following Sections.

Formulation of the World2 Control Problem

In the Section "Controlling the Market Growth Model", we formulated a control problem for the Market Growth model by removing one of the functional dependencies in the model, thereby freeing the variable. In this model, we choose a different approach.

We replace all table functions of the World2 model with their respective smooth spline interpolation counterparts. Typical for manual parameter finding, Forrester only changed the parameters once in the year 1970. In the optimization run, we will however remove the premise that the five policy parameters (see Table 2) only change once. We will consider them as time-dependent functions, with limitations on the speed of change as described in Section "Formulation of the Market Growth Control Problem".

Based on the considerations above, we formulate the following control problem based on the World2 model:

$$\begin{aligned}
\max \quad & \sum_t \Delta t \left(\text{qualityoflife}(T) \cdot \text{Population}(T) \right), && \text{(13a)} \\
\text{s.t.} \quad & \text{Control ranges defined in Table 2} && \text{(13b)} \\
& \text{Accumulated total costs}(T) \leq 3.8 \cdot 10^7, && \text{(13c)} \\
& \text{[model dynamics]}. && \text{(13d)}
\end{aligned}$$

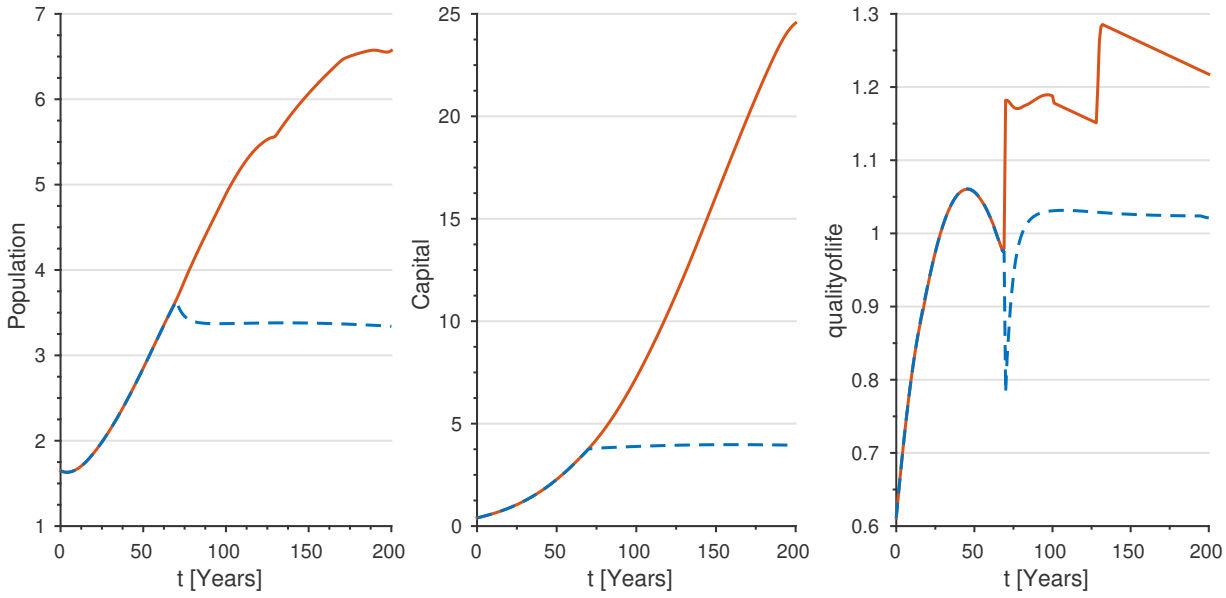
Optimization Results

As for the market growth model, we use CONOPT to solve the problem. The solving time for this model was 166 seconds, and the locally optimal objective value is 1047.74.

The results of the optimization are shown in Figures 9 and 10. In the three panels in Figure 9, a comparison between the base run and the optimization run for the state variables *Population*, *Capital*, and *quality of life* is shown. The last panel clearly shows, that in the optimization run, the quality of life is improved compared to the base run, at every point along the time axis. At the end of the simulation, the improvement amounts to roughly 20%. As shown in the first two panels, this improvement is accompanied by a continued growth of the population as well as the capital.

In Figure 10, the control solutions for the variables *Foodcoeff1970* and *Birthratenormal1970* are shown. The third panel compares the costs of the interventions accumulated at a given time. Since in the base run all parameters stay at the same value after the year 1970, it is clear that the cost incurred is the same at all times following the year 1970. While the cost of interventions is in the beginning slightly less than in the base run, there is a sharp peak of intervention, created by a strong intervention, reducing the birth rate in the year 2065.

All in all, the redistribution of the intervention towards different times and different control variables as in the base run has lead to a solution with a highly improved value of the objective function. To the best of our knowledge, no simulation of the World2 model was published so far that allowed for an ongoing growth of the population and did not lead to a collapse until the year 2100. As we demonstrated here, such a solution exists, if the premise of a constant parameter is replaced by time-varying parameters.



(a) (b) (c)

Figure 9: Comparison of variables of the World2 model in base run (dashed line) and optimized run (solid line).

Conclusions and Outlook

Mathematical optimization techniques for the numerical solution of nonlinear problems have advanced over the last years, so that they can be used to solve even more complex system dynamics optimization models. Besides parameter optimization, which is currently used in system dynamics software packages, this includes controls that are not single static changes at a certain point in time, but time-varying functions that come up with an individual control value for each time step. These are time-dependent functions that allow for a much more refined control of system dynamics models, which is not possible with the previous approach of manually setting parameters or using limited optimization techniques. Having such solutions at hand offers a more detailed view on the dynamic behavior of the model and its controllability. This can help modelers to gain new insights of their models, even when fully implementing the optimal policy values in a real situation, from which the simulation model originated, is not possible.

Existing numerical solvers such as CONOPT are able to solve such SDO problems to proven local optimality in very short time, typically less than one minute. The only technical obstacle is to approximate table functions by smooth functions, for which we suggest spline interpolation, because the numerical optimization software requires existing first and second derivates. We believe that most system dynamics models are actually smooth by their application domain ("nature does not jump"), and the existing non-

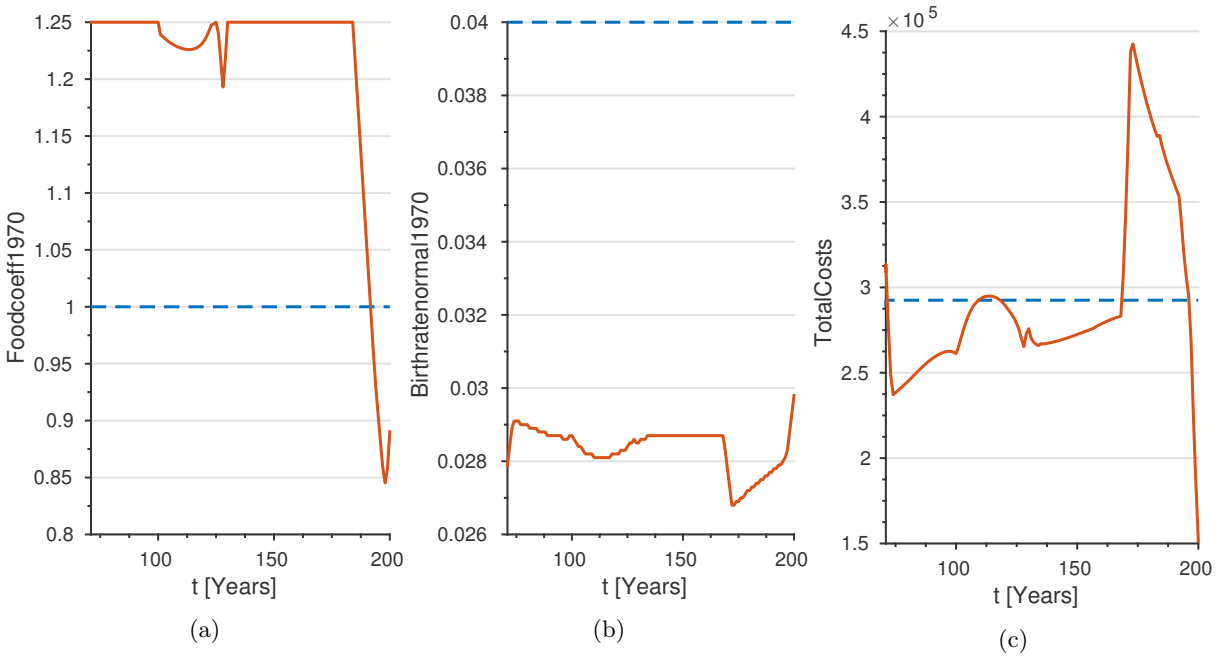


Figure 10: Comparison of control variables and intervention cost of the World2 model in base run (dashed line) and optimized run (solid line).

smoothness in published models is the result of the user interfaces of the most common software packages. Here, it is necessary to present the possibility of entering data not by filling out tables, but by directly drawing a spline curve, as for instance in CAD software packages.

Our research also has some further challenges for developers of numerical solvers: The solutions found by the nonlinear optimization techniques are only locally optimal, and no information is provided about how far they are away from a global optimal solution. To answer this question, one possible way is to embed the whole procedure in a branch-and-bound search that guarantees to find global optimal solutions in finite time. Recent research exists that addresses this direction.

Acknowledgement

We thank all software providers for sharing information and thus supporting our research.

References

Alborzi M. 2008. Augmenting System Dynamics with Genetic Algorithm and TOPSIS Multivariate Ranking Module for Multi-Criteria Optimization. In *The 12th International Conference of the System Dynamics Society, Stirling, Scotland.*

Artstein Z. 1980. Discrete and continuous bang-bang and facial spaces, or: Look for the extreme points. *SIAM Review*, 22(2):172–185.

Box MJ. 1965. A New Method of Constrained Optimization and a Comparison With Other Methods. *The Computer Journal*, 8(1):42 – 52.

Dangerfield B and Roberts C. 1996. An Overview of Strategy and Tactics in System Dynamics Optimization. *Journal of the Operational Research Society*, 47:405–423.

De Boor C. 1978. *A Practical Guide to Splines.* Number 27 in Applied Mathematical Sciences. Springer Verlag, New York.

Drud AS. 1985. CONOPT: A GRG Code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31(2):153 – 191.

Drud AS. 1994. A Large Scale GRG Code. *ORSA Journal on Computing*, 6(2):207 – 216.

Dynaplan AS. 2015. Dynaplan SMIA Software. Retrieved on June 30, 2015, from the website `http://dynaplan.com`.

Ford DN and Sterman JD. 1998. Dynamic modeling of product development processes. *System Dynamics Review*, 14(1):31–68.

Forio Corporation. 2015. /forio Software. Retrieved on June 30, 2015, from the website `http://forio.com`.

Forrester JW. 1968. Market Growth as Influenced by Capital Investment. *Industrial Management Review*, 9(2):83–105.

Forrester JW. 1969. *Urban Dynamics*. Waltham, MA, Pegasus Communications.

Forrester JW. 1971a. Counterintuitive Behavior of Social Systems. *Industrial Management Review*, 73(2):52–68.

Forrester JW. 1971b. *World Dynamics*. Wright-Allen Press, Boston, MA.

Fortmann-Roe S. 2014a. Insight Maker: A general-purpose tool for web-based modeling & simulation. *Simulation Modelling Practice and Theory*, 47:28–45. ISSN 1569190X. doi:10.1016/j.simpat.2014.03.013.

Fortmann-Roe S. 2014b. Insight Maker: A general-purpose tool for web-based modeling & simulation. *Simulation Modelling Practice and Theory*, 47:28–45.

Franco D. 2007. Fifty years of table functions. In *The 2007 International Conference of the System Dynamics Society and 50th Anniversay Celebration, Boston, Massachusetts, USA*. ISBN 978-0-9745329-8-1.

Fügenschuh A, Hayn C, and Michaels D. 2015. Mixed-Integer Linear Methods for Layout-Optimization of Screening Systems in Recovered Paper Production. *Optimization and Engineering*, 15(2):533–573.

GAMS Development Corporation. 2013. General Algebraic Modeling System (GAMS) Release 23.8.1. Washington, DC, USA.

Ghaffarzadegan N, Lyneis J, and Richardson GP. 2010. How small system dynamics models can help the public policy process. *System Dynamics Review*, 27(1):22–44.

GoldSim Technology Group. 2015. Goldsim Software. Retrieved on June 30, 2015, from the website `http://goldsim.com`.

Grösser SN. 2012. *Co-Evolution of Standards in Innovation Systems: The Dynamics of Voluntary and Legal Building Codes*. Springer Verlag, Heidelberg.

Grösser SN. 2014. Co-Evolution of Legal and Voluntary Standards: Development of Energy Efficiency in Swiss Residential Building Codes. *Technological Forecasting and Social Change*, 87(1):1–16.

Grösser SN and Tschupp R. 2015. Quality of system dynamics research: Evidence and recommendations for model-based studies. Technical report, submitted.

Homer J. 2014. Levels of evidence in system dynamics modeling. *System Dynamics Review*, 30(1-2):75–80.

isee systems, inc. 2015. STELLA Software. Retrieved on June 30, 2015, from the website `http://iseesystems.com`.

Jupp DLB. 1978. Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis*, 15(2):328–343. doi:10.1137/0715022.

Karush W. 1939. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. Master's thesis, Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois.

Kuhn H and Tucker W. 1951. Nonlinear Programming. In *Proceedings of 2nd Berkeley Symposium*, pages 481 – 492. Berkeley: University of California Press.

Morecroft JDW. 1985. Rationality in the analysis of behavioral simulation models. *Management Science*, 31(7):900–916. doi:10.1287/mnsc.31.7.900.

Morecroft JDW. 2012. Metaphorical Models for Limits to Growth and Industrialization. *Systems Research and Behavioral Science*, 29(6):645–666.

Morecroft JDW, Larsen ER, Lomi A, and Ginsberg A. 1995. The Dynamics of Resource Sharing - a Metaphorical Model. *System Dynamics Review*, 11(4):289–309.

Powell MJD. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7(2):155 – 162.

Powersim Software AS. 2015. Powersim Software. Retrieved on June 30, 2015, from the website `http://powersim.com`.

Rahmandad H and Repenning N. 2015. Capability Erosion Dynamics. *Strategic Management Journal*. Online available, to appear.

Rahmandad H and Sterman JD. 2012. Reporting guidelines for simulation-based research in social sciences. *System Dynamics Review*, 28(4):396–411.

Repenning NP, Goncalves P, and Black LJ. 2001. Past the Tipping Point: The Persistence of Firefighting in Product Development. *Industrial Management Review*, 43(4):44–54.

Richardson GP. 2011. Reflections on the foundations of system dynamics. *System Dynamics Review*, 27(3):219–243.

Sholtes RE. 1994. Optimizing System Behavior using Genetic Algorithms. In *The 12th International Conference of the System Dynamics Society, Stirling, Scotland*, pages 237–247.

Simon HA et al. 1984. Models of bounded rationality, volume 1: economic analysis and public policy. *MIT Press Books*, 1.

Sonneborn L and Van Vleck F. 1965. The Bang-Bang Principle for Linear Control Systems. *SIAM Journal of Control*, 2:151–159.

The AnyLogic Company. 2015. Anylogic Software. Retrieved on June 30, 2015, from the website `http://anylogic.com`.

Ventana Systems, Inc. 2015. Vensim Software. Retrieved on June 30, 2015, from the website `http://vensim.com`.

Vigerske S. 2013. *Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming*. Ph.D. thesis.

Wächter A and Biegler LT. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25 – 57.

Zhou W and Chen X. 2010. Global convergence of a new hybrid gauss-newton structured bfgs method for nonlinear least squares problems. *SIAM J. on Optimization*, 20(5):2422–2441. ISSN 1052-6234. doi:10.1137/090748470.